딥러닝 레포트 – 2

학과 : 컴퓨터공학과 학번 : 2019305061 이름 : 임주형

```
# 1-1
import numpy as np

A = np.array([[1, 3], [2, 3], [-1, 0]])
B = np.array([[0, 0, 2], [1, 4, -2], [3, 1, -1], [2, 0, 2]])
```

```
# 1-2

result = []

num1 = 0
num2 = 0
num3 = 0

for i in range(len(B)):
  for j in range(len(A[0])):
    num1 = B[i][0] * A[0][j]
    num2 = B[i][1] * A[1][j]
    num3 = B[i][2] * A[2][j]

    print(np.array([num1, num2, num3]))           # 각 배열의 요소를 곱한 결과
    result.append(np.array([num1, num2, num3]))

print(np.sum(result))
```

```
[ 0  0 -2]
[0 0 0]
[1 8 2]
[ 3 12  0]
[3 2 1]
[9 3 0]
[ 2  0 -2]
[6 0 0]
48
```

```python
# 2-1
import numpy as np

x = np.array([[0,0,0], [0,0,1], [0,1,0], [0,1,1], [1,0,0], [1,0,1], [1,1,0], [1,1,1]])
y_test = np.array([[1], [0], [1], [0], [0], [0], [1], [0]])
y_train = np.array([[0], [1], [1], [0], [1], [0], [0], [1]])
```

```python
# 2-2
import tensorflow as tf

model1 = tf.keras.Sequential([
    tf.keras.layers.Dense(units=30, activation='sigmoid', input_shape=(3,)),
    tf.keras.layers.Dense(units=20, activation='sigmoid'),
    tf.keras.layers.Dense(units=10, activation='sigmoid'),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])

model1.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.1), loss='mse')
model1.fit(x, y_train, epochs=50)
```

```
Epoch 40/50
1/1 [==============================] - 0s 10ms/step - loss: 0.2500
Epoch 41/50
1/1 [==============================] - 0s 14ms/step - loss: 0.2500
Epoch 42/50
1/1 [==============================] - 0s 12ms/step - loss: 0.2500
Epoch 43/50
1/1 [==============================] - 0s 12ms/step - loss: 0.2500
Epoch 44/50
1/1 [==============================] - 0s 11ms/step - loss: 0.2500
Epoch 45/50
1/1 [==============================] - 0s 11ms/step - loss: 0.2500
Epoch 46/50
1/1 [==============================] - 0s 9ms/step - loss: 0.2500
Epoch 47/50
1/1 [==============================] - 0s 10ms/step - loss: 0.2500
Epoch 48/50
1/1 [==============================] - 0s 10ms/step - loss: 0.2500
Epoch 49/50
1/1 [==============================] - 0s 10ms/step - loss: 0.2500
Epoch 50/50
1/1 [==============================] - 0s 11ms/step - loss: 0.2500
<keras.src.callbacks.History at 0x7cb7924c49d0>
```

```
# 2-2
model2 = tf.keras.Sequential([
    tf.keras.layers.Dense(units=100, activation='relu', input_shape=(3,)),
    tf.keras.layers.Dense(units=50, activation='relu'),
    tf.keras.layers.Dense(units=1, activation='relu')
])

model2.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.1), loss='mse')
model2.fit(x, y_train, epochs=50)
```

```
Epoch 40/50
1/1 [==============================] - 0s 12ms/step - loss: 0.0857
Epoch 41/50
1/1 [==============================] - 0s 8ms/step - loss: 0.0835
Epoch 42/50
1/1 [==============================] - 0s 11ms/step - loss: 0.0813
Epoch 43/50
1/1 [==============================] - 0s 9ms/step - loss: 0.0793
Epoch 44/50
1/1 [==============================] - 0s 11ms/step - loss: 0.0772
Epoch 45/50
1/1 [==============================] - 0s 10ms/step - loss: 0.0750
Epoch 46/50
1/1 [==============================] - 0s 9ms/step - loss: 0.0732
Epoch 47/50
1/1 [==============================] - 0s 11ms/step - loss: 0.0704
Epoch 48/50
1/1 [==============================] - 0s 9ms/step - loss: 0.0684
Epoch 49/50
1/1 [==============================] - 0s 9ms/step - loss: 0.0659
Epoch 50/50
1/1 [==============================] - 0s 9ms/step - loss: 0.0647
<keras.src.callbacks.History at 0x7cb79c1ab580>
```

```
# 2-3
model1.evaluate(x, y_test, verbose=2)

model2.evaluate(x, y_test, verbose=2)
```

```
1/1 - 0s - loss: 0.2507 - 19ms/epoch - 19ms/step
1/1 - 0s - loss: 0.3698 - 22ms/epoch - 22ms/step
0.3698039948940277
```

```python
# 3-1

import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np

fashion_mnist = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

class_name = ['T-shirt', 'Trouser', 'Pullover', 'Dress', 'Coat',
              'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

x_train, x_test = x_train/ 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28,28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0, 2),
    tf.keras.layers.Dense(10, activation='softmax'),
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
Epoch 1/5
1875/1875 [==============================] - 17s 8ms/step - loss: 0.4988 - accuracy: 0.8239
Epoch 2/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3740 - accuracy: 0.8658
Epoch 3/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3350 - accuracy: 0.8771
Epoch 4/5
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3135 - accuracy: 0.8851
Epoch 5/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2947 - accuracy: 0.8904
<keras.src.callbacks.History at 0x79676317ad40>
```

```python
# 이미지 및 예측 출력
import random

plt.figure(figsize=(10, 10))
# random_index = random.randint(0, len(y_train))
for i in range(5):
    random_index = random.randint(0, len(y_train))
    plt.subplot(5, 5, i+1)
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_train[random_index], cmap=plt.cm.binary)
    predicted_label = class_name[model.predict(x_train[random_index:random_index+1]).argmax()]
    # print(y_train[random_index])
    true_label = class_name[y_train[random_index]]
    plt.title(predicted_label)
plt.show()
```

```
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
```


```python
# 3-2

import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np

fashion_mnist = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

class_name = ['T-shirt', 'Trouser', 'Pullover', 'Dress', 'Coat',
              'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 1s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

```
x_train_final = x_train.reshape((-1, 28, 28, 1)) / 255.
x_test_final = x_test.reshape((-1, 28, 28, 1)) / 255.

model_with_conv = tf.keras.Sequential([
    tf.keras.layers.Conv2D(96, (3,3), padding='same', activation='relu',
                           input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D((2,2), strides=2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model_with_conv.compile(optimizer='adam',
                        loss='sparse_categorical_crossentropy',
                        metrics=['accuracy'])

model_with_conv.fit(x_train_final, y_train, epochs=5)
model_with_conv.fit(x_test_final, y_test, verbose=2)
```

```
Epoch 1/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3651 - accuracy: 0.8694
Epoch 2/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2414 - accuracy: 0.9113
Epoch 3/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1963 - accuracy: 0.9273
Epoch 4/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1618 - accuracy: 0.9395
Epoch 5/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1318 - accuracy: 0.9511
313/313 - 1s - loss: 0.2546 - accuracy: 0.9108 - 1s/epoch - 4ms/step
<keras.src.callbacks.History at 0x7f918386ce20>
```

```
x_train_final = x_train.reshape((-1, 28, 28, 1)) / 255.
x_test_final = x_test.reshape((-1, 28, 28, 1)) / 255.

model_with_conv2 = tf.keras.Sequential([
    tf.keras.layers.Conv2D(128, (3,3), padding='same', activation='relu',
                           input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D((3,3), strides=2),
    tf.keras.layers.Conv2D(256, (5,5), padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D((1,1), strides=2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model_with_conv2.compile(optimizer='adam',
                         loss='sparse_categorical_crossentropy',
                         metrics=['accuracy'])

model_with_conv2.fit(x_train_final, y_train, epochs=5)
model_with_conv2.fit(x_test_final, y_test, verbose=2)
```

```
Epoch 1/5
1875/1875 [==============================] - 13s 6ms/step - loss: 0.3615 - accuracy: 0.8677
Epoch 2/5
1875/1875 [==============================] - 12s 6ms/step - loss: 0.2369 - accuracy: 0.9127
Epoch 3/5
1875/1875 [==============================] - 12s 6ms/step - loss: 0.1906 - accuracy: 0.9284
Epoch 4/5
1875/1875 [==============================] - 12s 6ms/step - loss: 0.1579 - accuracy: 0.9406
Epoch 5/5
1875/1875 [==============================] - 12s 6ms/step - loss: 0.1300 - accuracy: 0.9514
313/313 - 2s - loss: 0.2297 - accuracy: 0.9198 - 2s/epoch - 7ms/step
<keras.src.callbacks.History at 0x7b645d32a3e0>
```

```
x_train_final = x_train.reshape((-1, 28, 28, 1)) / 255.
x_test_final = x_test.reshape((-1, 28, 28, 1)) / 255.

model_with_conv3 = tf.keras.Sequential([
    tf.keras.layers.Conv2D(128, (3,3), padding='same', activation='relu',
                           input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D((2,2), strides=2),
    tf.keras.layers.Conv2D(256, (5,5), padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D((2,2), strides=2),
    tf.keras.layers.Conv2D(256, (3,3), padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D((1,1), strides=2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model_with_conv3.compile(optimizer='adam',
                         loss='sparse_categorical_crossentropy',
                         metrics=['accuracy'])

model_with_conv3.fit(x_train_final, y_train, epochs=5)
model_with_conv3.fit(x_test_final, y_test, verbose=2)
```

```
Epoch 1/5
1875/1875 [==============================] - 15s 7ms/step - loss: 0.3784 - accuracy: 0.8614
Epoch 2/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.2398 - accuracy: 0.9122
Epoch 3/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.1920 - accuracy: 0.9291
Epoch 4/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.1594 - accuracy: 0.9406
Epoch 5/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.1322 - accuracy: 0.9501
313/313 - 3s - loss: 0.2369 - accuracy: 0.9155 - 3s/epoch - 8ms/step
<keras.src.callbacks.History at 0x7b63ec44abc0>
```

```python
# 3-3

import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np

fashion_mnist = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

x_train_final = x_train.reshape((-1, 28, 28, 1)) / 255.
x_test_final = x_test.reshape((-1, 28, 28, 1)) / 255.
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 2s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 1s 0us/step
```

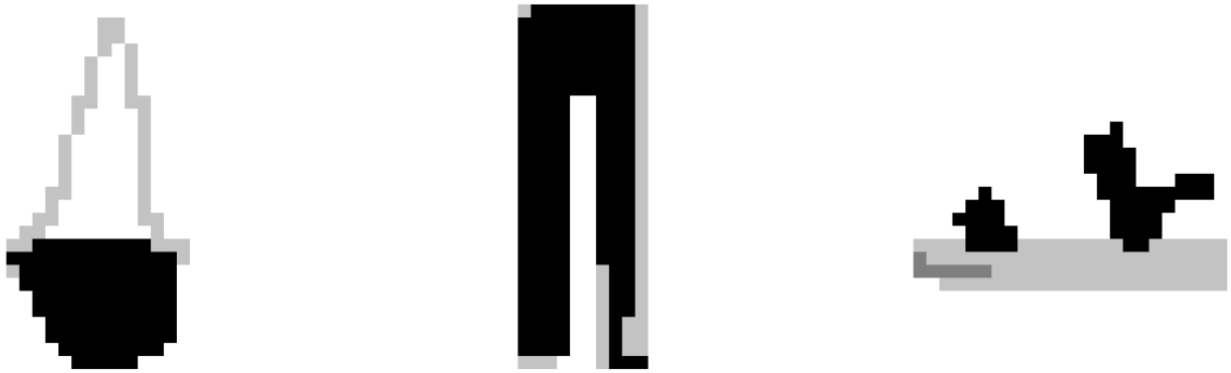
```python
from google.colab import files

uploaded = files.upload()
```

파일 선택 파일 3개

- **Bag.png**(image/png) - 250 bytes, last modified: 2023. 10. 28. - 100% done
- **Trouser.png**(image/png) - 196 bytes, last modified: 2023. 10. 28. - 100% done
- **샌들.png**(image/png) - 268 bytes, last modified: 2023. 10. 28. - 100% done

```
Saving Bag.png to Bag (3).png
Saving Trouser.png to Trouser (3).png
Saving 샌들.png to 샌들 (4).png
```

```python
# 그린 그림 출력
import cv2
import matplotlib.pyplot as plt

# 업로드한 이미지 경로
image_path_1 = list(uploaded.keys())[0]
image_path_2 = list(uploaded.keys())[1]
image_path_3 = list(uploaded.keys())[2]

# 이미지 전처리
img1 = cv2.imread(image_path_1, cv2.IMREAD_GRAYSCALE)
img2 = cv2.imread(image_path_2, cv2.IMREAD_GRAYSCALE)
img3 = cv2.imread(image_path_3, cv2.IMREAD_GRAYSCALE)

# 이미지를 28x28로 크기 조절
resized_img1 = cv2.resize(img1, (28, 28), interpolation=cv2.INTER_LINEAR)
resized_img2 = cv2.resize(img2, (28, 28), interpolation=cv2.INTER_LINEAR)
resized_img3 = cv2.resize(img3, (28, 28), interpolation=cv2.INTER_LINEAR)

# 이미지를 1차원 배열로 변환 및 정규화
resized_img1 = resized_img1.reshape((28, 28, 1)) / 255.0
resized_img2 = resized_img2.reshape((28, 28, 1)) / 255.0
resized_img3 = resized_img3.reshape((28, 28, 1)) / 255.0

img_list = [resized_img1, resized_img2, resized_img3]

plt.figure(figsize=(15, 4))  # 그림 크기를 적절하게 조절

for i in range(len(img_list)):
    plt.subplot(1, len(img_list), i + 1)
    plt.imshow(img_list[i].reshape(28, 28), cmap='gray')
    plt.axis('off')

plt.show()
```



```
from sklearn.metrics import accuracy_score
from PIL import Image

image_fil1 = list(uploaded.keys())[0]
image_fil2 = list(uploaded.keys())[1]
image_fil3 = list(uploaded.keys())[2]

image1 = Image.open(image_fil1)
image2 = Image.open(image_fil2)
image3 = Image.open(image_fil3)

image1 = image1.resize((28, 28))
image2 = image2.resize((28, 28))
image3 = image3.resize((28, 28))

image1 = image1.convert("L")
image2 = image2.convert("L")
image3 = image3.convert("L")

image_array1 = np.array(image1)
image_array2 = np.array(image2)
image_array3 = np.array(image3)

image_array1 = image_array1.reshape((1, 28, 28, 1))
image_array2 = image_array2.reshape((1, 28, 28, 1))
image_array3 = image_array3.reshape((1, 28, 28, 1))

predicted_label1 = model_with_conv1.predict(image_array1)
predicted_label2 = model_with_conv1.predict(image_array2)
predicted_label3 = model_with_conv1.predict(image_array3)
```

```
predicted_class1 = np.argmax(predicted_label1)
predicted_class2 = np.argmax(predicted_label2)
predicted_class3 = np.argmax(predicted_label3)

predicted_classes = [predicted_class1, predicted_class2, predicted_class3]
true_label = [0, 0, 0]
index = 0

for pc in predicted_classes:
  for y_label in y_test:
    if pc == y_label: # 예측한 레이블과 실제 레이블이 같은지 확인
      true_label[index] = 1.0          # 같다면 정답 레이블 1로 변경
      break
  index += 1

sum = 0
for a in true_label:
  sum += a
result = sum / 3
print(predicted_classes)
print("1쌍 모델 결과: {}".format(result))
```

```
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 40ms/step
[8, 5, 8]
1쌍 모델 결과: 1.0
```

```
predicted_label1 = model_with_conv2.predict(image_array1)
predicted_label2 = model_with_conv2.predict(image_array2)
predicted_label3 = model_with_conv2.predict(image_array3)

predicted_class1 = np.argmax(predicted_label1)
predicted_class2 = np.argmax(predicted_label2)
predicted_class3 = np.argmax(predicted_label3)

predicted_classes = [predicted_class1, predicted_class2, predicted_class3]
true_label = [0, 0, 0]
index = 0

for pc in predicted_classes:
  for y_label in y_test:
    if pc == y_label: # 예측한 레이블과 실제 레이블이 같은지 확인
      true_label[index] = 1.0          # 같다면 정답 레이블 1로 변경
      break
  index += 1

sum = 0
for a in true_label:
  sum += a
result = sum / 3
print(predicted_classes)
print("2쌍 모델 결과: {}".format(result))
```

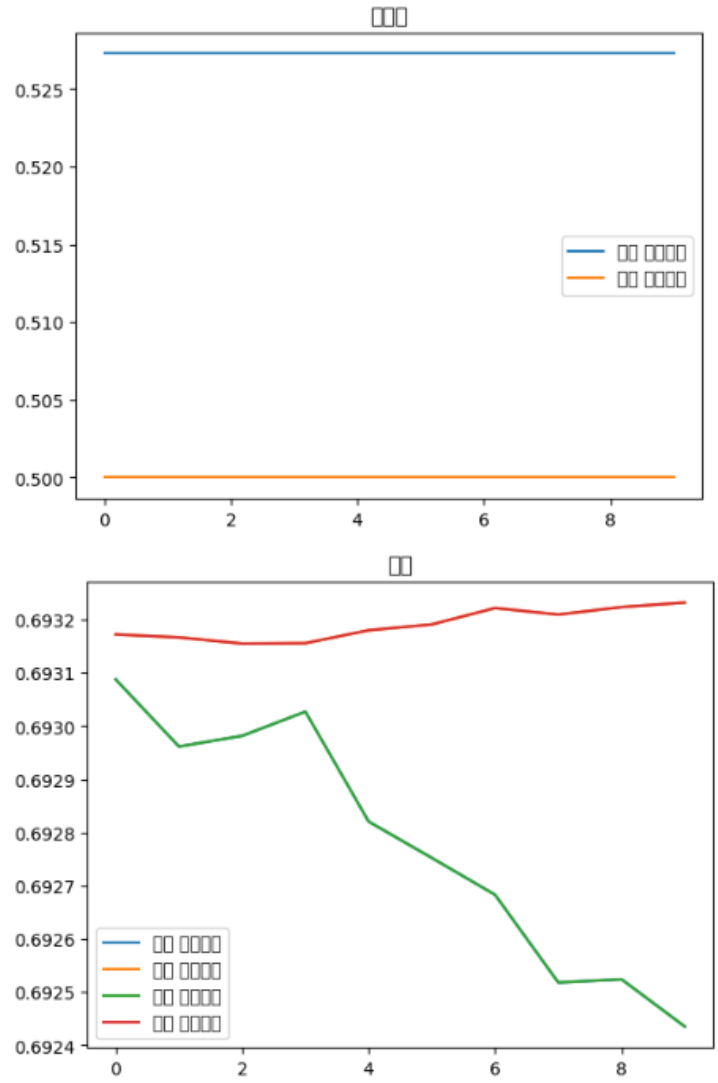
```
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 51ms/step
[8, 8, 8]
2쌍 모델 결과: 1.0
```

```
predicted_label1 = model_with_conv3.predict(image_array1)
predicted_label2 = model_with_conv3.predict(image_array2)
predicted_label3 = model_with_conv3.predict(image_array3)

predicted_class1 = np.argmax(predicted_label1)
predicted_class2 = np.argmax(predicted_label2)
predicted_class3 = np.argmax(predicted_label3)

predicted_classes = [predicted_class1, predicted_class2, predicted_class3]
true_label = [0, 0, 0]
index = 0

for pc in predicted_classes:
  for y_label in y_test:
    if pc == y_label: # 예측한 레이블과 실제 레이블이 같은지 확인
      true_label[index] = 1.0          # 같다면 정답 레이블 1로 변경
      break
  index += 1

sum = 0
for a in true_label:
  sum += a
result = sum / 3
print(predicted_classes)
print("3쌍 모델 결과: {}".format(result))
```

```
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 26ms/step
[8, 8, 8]
3쌍 모델 결과: 1.0
```

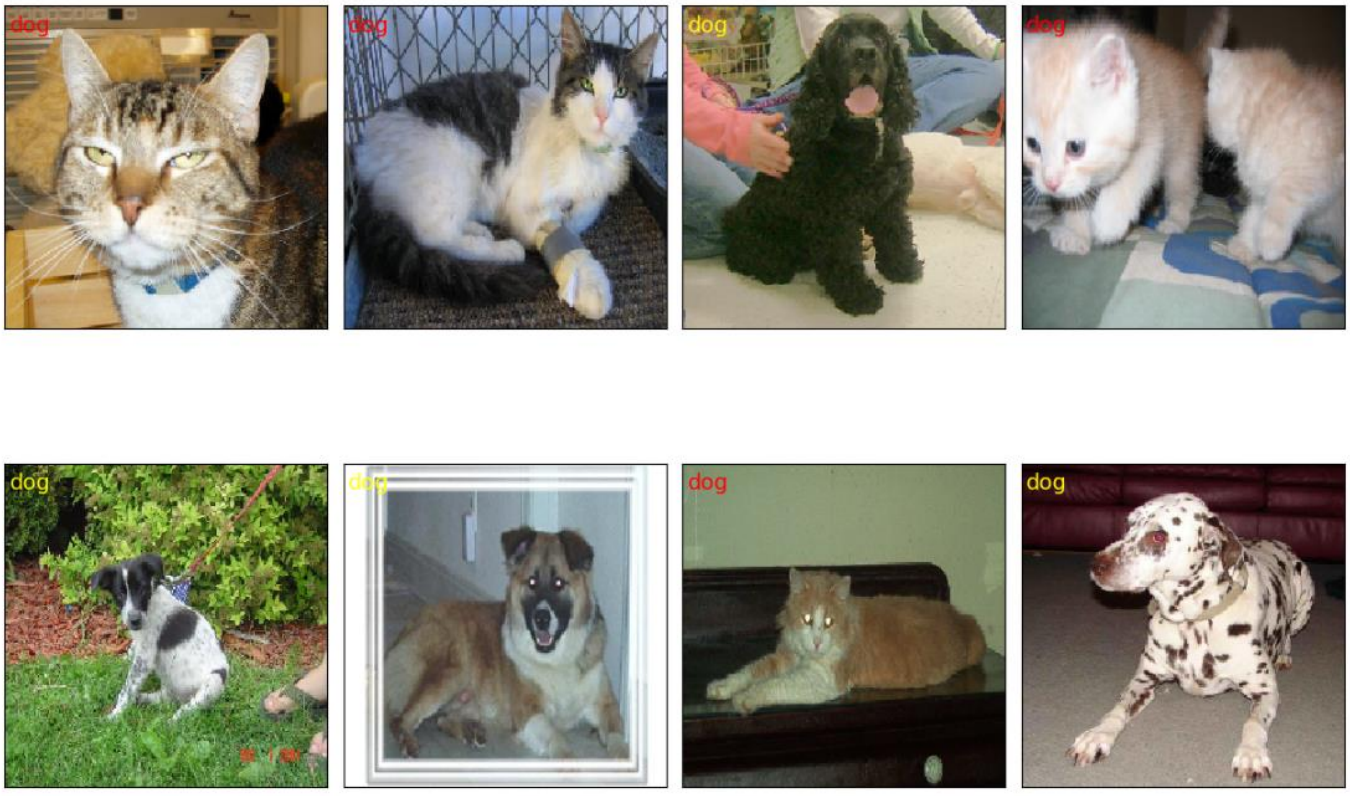
```
# 5-18코드
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(accuracy))

plt.plot(epochs, accuracy, label="훈련 데이터셋")
plt.plot(epochs, val_accuracy, label="검증 데이터셋")
plt.legend()
plt.title('정확도')
plt.figure()


plt.plot(epochs, loss, label="훈련 데이터셋")
plt.plot(epochs, val_loss, label="검증 데이터셋")
plt.legend()
plt.title('오차')


plt.plot(epochs, loss, label="훈련 데이터셋")
plt.plot(epochs, val_loss, label="검증 데이터셋")
plt.legend()
plt.title('오차')
```

```
Text(0.5, 1.0, '오차')
```



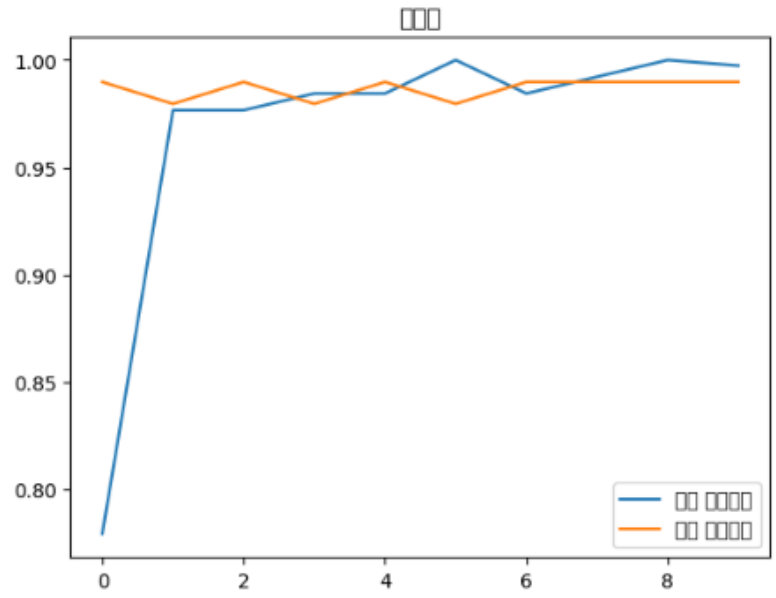
```
# 5-19코드
class_names = ['cat', 'dog']
validation, label_batch  = next(iter(valid_generator))
prediction_values = model.predict(validation)
prediction_values = np.argmax(prediction_values, axis=1)

fig = plt.figure(figsize=(12, 8))
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)

for i in range(8):
    ax = fig.add_subplot(2, 4, i + 1, xticks=[], yticks=[])
    ax.imshow(validation[i,:],cmap=plt.cm.gray_r, interpolation='nearest')
    if prediction_values[i] == np.argmax(label_batch[i]):
        ax.text(3, 17, class_names[prediction_values[i]], color='yellow', fontsize=14)
    else:
        ax.text(3, 17, class_names[prediction_values[i]], color='red', fontsize=14)
```

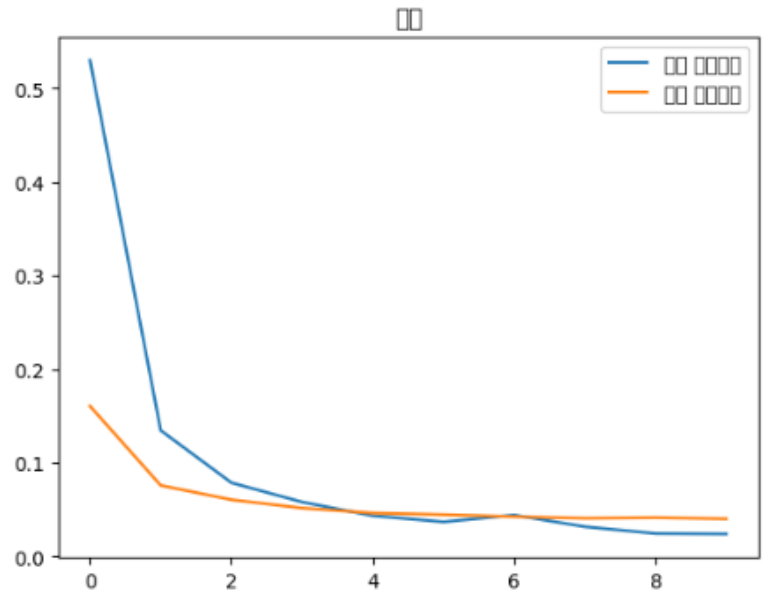
```
1/1 [==============================] - 1s 1s/step
```



```
# 5-30코드
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(accuracy))

plt.plot(epochs, accuracy, label="훈련 데이터셋")
plt.plot(epochs, val_accuracy, label="검증 데이터셋")
plt.legend()
plt.title('정확도')
plt.figure()

plt.plot(epochs, loss, label="훈련 데이터셋")
plt.plot(epochs, val_loss, label="검증 데이터셋")
plt.legend()
plt.title('오차')
```

```
Text(0.5, 1.0, '오차')
```





```
# 5-31코드
class_names = ['cat', 'dog']
validation, label_batch  = next(iter(valid_generator))
prediction_values = model.predict(validation)
prediction_values = np.argmax(prediction_values, axis=1)

fig = plt.figure(figsize=(12, 8))
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)

for i in range(8):
    ax = fig.add_subplot(2, 4, i + 1, xticks=[], yticks=[])
    ax.imshow(validation[i,:],cmap=plt.cm.gray_r, interpolation='nearest')
    if prediction_values[i] == np.argmax(label_batch[i]):
        ax.text(3, 17, class_names[prediction_values[i]], color='yellow', fontsize=14)
    else:
        ax.text(3, 17, class_names[prediction_values[i]], color='red', fontsize=14)
```

```
1/1 [==============================] - 3s 3s/step
```